To achieve the goals of e-Science, we must change research culture globally

To achieve the goals of e-Science, we must change research culture globally

Neil Chue Hong

Introduction

An ambitious plan

In the early 2000s the UK's research funders embarked on an ambitious initiative that brought together domain scientists, computer scientists, and software engineers in collaboration to solve grand challenges of research. As coined by John Taylor, at that time the Director General of the UK's Office of Science and Technology, this £120m "e-Science" program invested in large scale, collaborative projects to develop infrastructure, software and resources to support researchers working in all areas of research [18].

This approach was picked up by other countries, including the United States of America through the National Science Foundation's Office of Cyber-Infrastructure [17] and The Netherlands through The Netherlands e-Science Center[1]. Although e-Science is funded differently across the world, it represents a paradigm shift, identifying the need for infrastructure and tools to support data-intensive research workflows, supporting the case for open data and open science to make research more efficient, and supporting research integrity as experiments become more complex.

Software as a shaky foundation

Underpinning this is software: if e-Science is defined as the use of computational tools and platforms to enable novel research, almost all of modern research is e-Science. This is evidenced by surveys conducted of UK researchers at research-intensive universities in 2014 [11] (Fig. 1) and US post-doctoral researchers in 2017 [16]. These found that the majority of respondents (69 % in UK, 63 % in US) would find it impossible to conduct their research without software.

A key challenge identified in these surveys is that a large proportion of researchers (54 % in US; 45 % in UK, including 21 % of those who develop their own research software) had received no training in software development, despite researchers often having to extend or modify the software they are using. This was backed by a 2014 study [3] by the UK Biotechnology and Biological Sciences Research Council and Medical Research Council on vulnerable skills, which noted that "informatics skills are applicable to many areas of both the biosciences and the medical sciences" but "data analytics, especially bioinformatics, appear to be particularly vulnerable." Without the correct skills and experience, the threat is that research may be reaching a point where researchers are conducting experiments without truly knowing what is happening.

This crisis of "reproducibility" – the ability to take the data and methodology (often in the form of software) and reproduce the results published in a paper – is of particular concern given the implications it can have in certain fields. A 2012 study by Begley and Ellis [4] reviewed a decade of landmark

[1] https://www.esciencecenter.nl/.

https://doi.org/10.1007/s00287-018-01134-1

Neil Chue Hong
Software Sustainability Institute, EPCC, University of Edinburgh,
Edinburgh, United Kingdom
E-Mail: N.ChueHong@software.ac.uk

Published online: 13 December 2018

Abstract

The e-Science program was initiated in the United Kingdom in the early 2000s with the aim of bringing together researchers in large scale, collaborative projects involving software and computation to solve grand challenges. A legacy of this program has been an understanding of the importance of the people behind the software, the researchers and research software engineers, as well as the challenges of developing and maintaining code that is reusable given the problems of software decay.

The Software Sustainability Institute was established in the UK to provide support and direction for the research software community through consultancy, training, engagement, and policy campaigns. Through this it has worked with an international community of collaborators, in the UK, in Europe, and across the world to support reusability, research integrity, and transparency, recognizing that to achieve the goals of e-Science, we must change research culture globally.

cancer research papers and found that 47 out of 53 were irreproducible, often for simple-to-correct reasons such as failure to repeat experiments, failure to show all data, and inappropriate use of statistical tests. Each of these issues should benefit from the increased use of computational techniques, and yet similar failures can be seen happening in disciplines such as genetics [13] and computer science [6], where even getting hold of the code used – let alone verifying its correctness – was an issue.

A culture of challenges

The question is why this culture exists. Is it because of the pressures of "publish or perish" rewarding speed and quantity over rigor and quality? Certainly, this plays a part, as discussions at the Software Sustainability Institute's Collaborations Workshops[2] have noted. Because much code is written by students and early career postdocs, the emphasis is getting the software only to the point where it can be used to produce results for a paper. Additionally, because academic reviewers (both of research papers and research grants) favor novelty, there is no motivation to go back and make software more reusable or robust.

A big issue is that whilst it is believed that there are many errors introduced by software in the literature, there is no incentive to fix them. The chances of an author being found out diminish, if their code is not published, and for those who do notice an error and try to correct it, the dreaded "retraction" is often the only way to do it [10, 15]. There are discussions amongst journals and publishers to change this [9], and it will be necessary to change the culture around scientific integrity and correctness before software quality improves across all codes.

Finally, software is hard to work with. Forget making research reproducible, it is hard enough to use it 6 months later. Konrad Hinsen has coined the term "software collapse" [12] due to the fact that software stops working eventually if is not actively maintained. Hinsen notes that software stacks used in computational science have a nearly universal multi-layer structure:

- Project-specific software: whatever it takes to do a computation using software building blocks from the lower three levels; scripts, workflows, computational notebooks, small special-purpose libraries, and utilities.
- Discipline-specific research software: tools and libraries that implement models and methods which are developed and used by research communities.
- Scientific infrastructure: libraries, and utilities used for research in many different disciplines, such as LAPACK, NumPy, or Gnuplot.
- Non-scientific infrastructure: operating systems, compilers, and support code for I/O, user interfaces, etc.

where software in each layer builds on and depends on software in all layers below it, and any changes in any lower layer can cause it to collapse.

The reproducible research community has traditionally focused on the project-specific software layer, where the main obstacle (as described earlier) is the unavailability of the code. This is also the software that receives the least attention after a project ends – it may become the starting point for software specific to another project – but it is rarely used by anyone outside the project that developed it. Again, it is down to incentive for improvement, for reusing

[2] https://www.software.ac.uk/workshops.

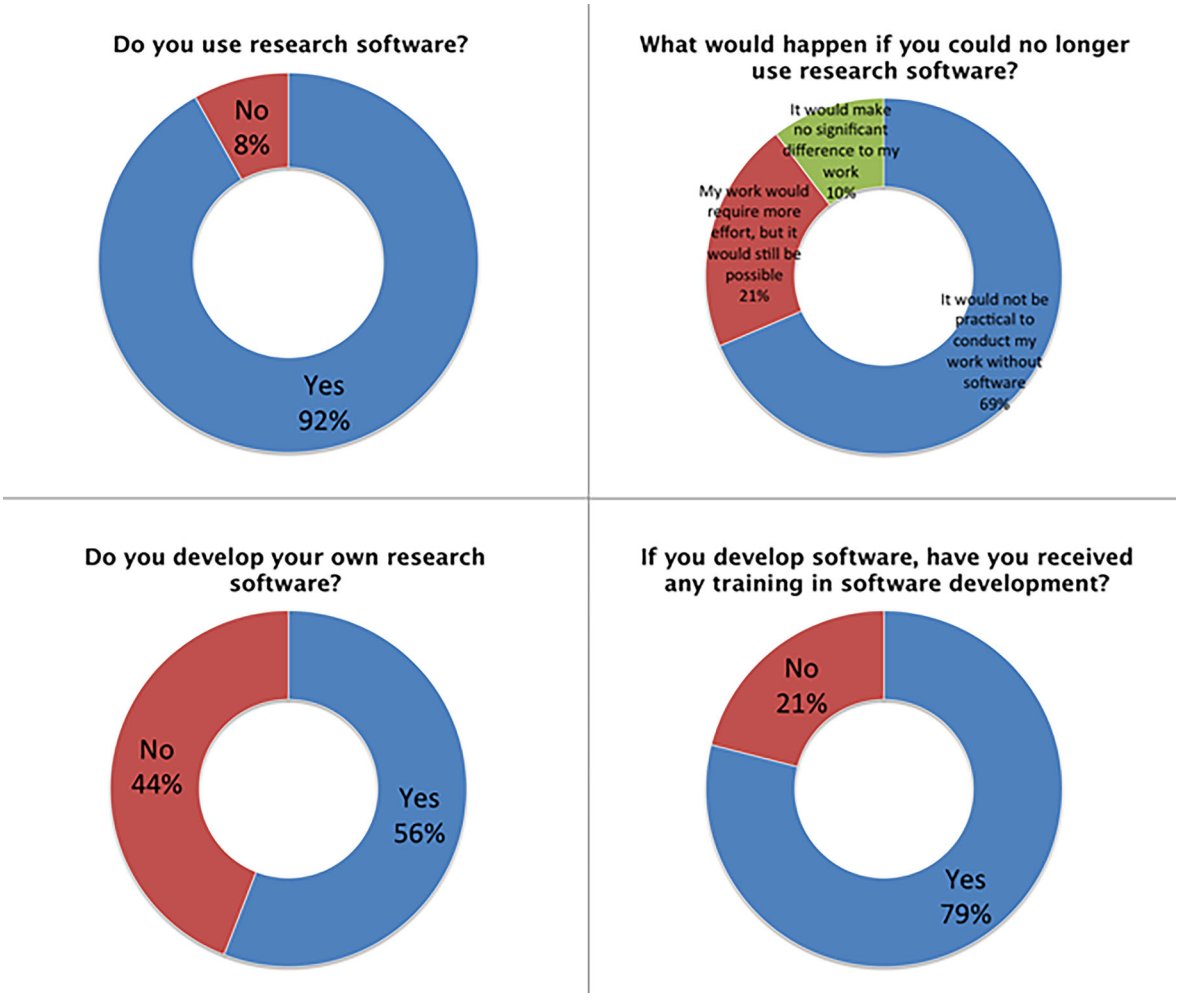

Fig. 1 UK Research Software Survey Results (2014). doi: 10.5281/zenodo.14809

others' work, for contributing back to the "communal maintenance". Whilst some software achieves this sustainability, making its way perhaps to Hinsen's lower levels of discipline-specific research software or scientific infrastructure, the majority is discarded because of the effort to keep it running. This is not necessarily a problem – much of this software is used to solve a single, specific problem – but it is an issue if results are to be revisited later, when even if the code has been made available, it cannot be easily run because of changes in the underlying libraries, operating systems, and other dependencies. Although virtual machines and, more recently, containers have been posited as solutions for this problem of decay, in reality, they suffer the same issues. C. Titus Brown, a regular commentator in this area, concludes [5] that "on a decadal time scale, we cannot rely on software to run repeatably." Daniel S. Katz, who was previously NSF's Program Director for their Software Infrastructure for Sustained Innovation program, asks "Is software reproducibility possible and practical?" [14]. This pushback against the vision of perfect reproducibility is perhaps the most important debate in the research software community at present, because it forces us to ask the question about the trade-off between the costs of making research results reproducible, the usefulness of doing so for a particular research output, and the potential new research that could be accomplished with those resources.

In the end, perhaps the pragmatic approach is to make it easier (not easy) to inspect research. After all, the consequences of mistakes in software can be large. In the case of economists Carmen Reinhart

and Kenneth Rogoff, their research paper "Growth in a Time of Debt" [19] was used as evidence to support the implementation of austerity measures by several governments, including in the UK. However, a software error in an Excel spreadsheet – identified by a student [7] – meant that the results were not as conclusive as they first appeared. However, this discovery could not have happened if Reinhart and Rogoff had not been willing to provide their data and code to the student to analyze. Perhaps open science is the way to go for practical reproducibility?

The work of the Software Sustainability Institute

Building a cross-disciplinary community

The Software Sustainability Institute was set up [8] in 2010 as a partnership between the universities of Edinburgh, Manchester, Oxford, and Southampton. Its goal is to cultivate world-class research with software, by overcoming the problems that beset research software and changing the way that researchers view it; a way to address the challenges identified in the previous section.

Over the last 8 years, it has evolved (with funding from the Engineering and Physical Sciences Research Council, Economic and Social Sciences Research Council, and Biotechnology and Biological Sciences Research Council) to bring the community together in tackling these problems. It works on several scales, providing consultancy and advice direct to researchers who are developing software, enabling researchers across the UK to access appropriate training, and running events and campaigns to support and facilitate champions of research culture change through best practice [23] and policy [1].

This work is inherently cross-disciplinary. The Institute works with researchers from every discipline. Although individual requirements are different, many of the issues faced are the same, including software reuse, reproducibility, and accumulation of software technical debt. To understand how to address these and other issues, an understanding of them must first be built from within the research community.

The Institute's Fellowship Program [20] is a cost-effective approach to obtain community intelligence, recruiting members of the software research community in a way that benefits them and the Institute. The program provides bursaries to researchers across many domains, institutions, and career stages in exchange for their expertise and advice, although it focuses on early career researchers. Institute Fellows receive funding for attending conferences and workshops that focus on different aspects of software development and use in their research area. In exchange, they supply the domain intelligence used to develop wider policy objectives. As domain specialists, they are also ideally placed to identify other opportunities (such as consultancy projects) and they also provide feedback on activities undertaken by the Institute. Over 5 years, the Institute has created a network of more than 100 Fellows, spanning subjects including history, engineering, imaging and earth sciences, to name a few.

Given the limited budget, the Fellowship Program represents a significant return on investment, with the Fellows having established a keen group identity, vision, and set of activities that assist the UK research community in many areas. A remarkable outcome of the program is that many Fellows go beyond their initial remit, taking initiative in organizing events within their respective communities, delivering presentations, advocating best practices, and suggesting other activities that have relevance and benefit to the Institute. A major event organized and attended by four Fellows was a "Software & Research Town Hall Meeting" at the American Geophysical Union (AGU) Fall Meeting; this is one of the AGU's largest community events, attended by 22,000 researchers from all over the world. Fellows have also led events that have focused on addressing the software development skills gap and discussing cultural issues. For example, in coordination with the training team, our Fellows have been instrumental in the growth of Software Carpentry [22] and Data Carpentry[3] in the UK, with two of them becoming fully qualified instructor trainers and having been involved in the creation of new training as part of Library Carpentry[4]. As the program has progressed, our Fellows have also progressed to prominent positions, including Head of Research Services at the British Library, Director of Research Engineering at the Alan Turing Institute, and Vice President (Promotion of Computing) of the British Computer Society, all helping to influence policy and promote the objectives of the Institute.

[3] http://www.datacarpentry.org/.
[4] https://librarycarpentry.github.io/.

Recognizing the right people

The Institute also aims to bring together the right people to understand and address topical issues. From the flagship Collaborations Workshop unconference event [21] to workshops on themes such as software credit, measuring the impact of workshops, and research data visualization, our aim is to ensure that the widest possible input is sought for key software challenges facing research. This has particularly been the case with two topics: recognition of research software engineers and computational skills for researchers.

In 2012, following a discussion at the Collaborations Workshop 2012, a number of attendees published a paper (including this author) on "The research software engineer" [2] which noted that:

"Computational work must reflect the committed attitude of experimentalists towards caring about precise, professional, repeatable, meticulous work – no-one with the same casual attitude to experimental instrumentation as many researchers have to code would be allowed anywhere near a lab."

This returned to the themes of e-Science, and in particular to the idea that research should be carried out by teams of varied specialists, including individuals who "combine a professional attitude to the exercise of software engineering with a deep understanding of research topics" – *research software engineers*. The paper made three recommendations to address the challenges identified:

- "The REF [Research Evaluation Framework for assessing the quality of research in UK institutions] allows for recognition of software deliverables through its system for impact measurement. In practice, however, this depends on the decisions of individual panels. REF panels should be given clear guidance on the importance of weighing software as a 'first class' research output.
- Research software engineers are a new role in academic institutions. Institutions and funding panels should recognize the value of this role in funding research proposals and in providing career progression and succession for such individuals.
- All researchers must be exposed to best practices in software development. The fundamentals of good software engineering should form part of every researcher's basic training."

The first is still in progress and awaits the imminent publication of guidance for the 2021 REF assessment process. The third has been achieved through the success of the international carpentries open training initiative, for whom the Institute is the UK coordinator. Since the first event in 2012, over 120 workshops have been run in the UK, teaching thousands of researchers the basic software development (through software carpentry) and data management and analysis (through data carpentry) skills they require to conduct research professionally. More than this, there are now over 100 trained instructors spread across the UK (Fig. 2), ready to pass on their knowledge to new researchers and students.

Finally, the second point, on recognition for research software engineers, has become one of the success stories of the last years. The initial paper sparked a debate into the role, quickly followed by the setting up of the UK Research Software Engineer Association[5] and recognition by the Engineering and Physical Science Research Council through the funding of RSE Fellowships – putting the position on par with academics. Since then, many Research Software Engineering groups have been set up at universities, providing services and support to researchers developing code, and changing the business of research software. These groups are actively involved in funding proposals, ensuring that the effort required to help address Hinsen's software decay is properly understood and costed. 5 years since the first discussions, and a second round of EPSRC RSE Fellowships have been awarded, job adverts for RSEs are posted every week; the RSE Association is about to incorporate into a legal professional body, and chapters have opened in The Netherlands, Canada, South Africa, and, of course, Germany[6].

e-Science is international

It seems obvious, but the main finding that the Institute has learned from working with researchers in the UK is that research is global. Not only are research teams at UK universities drawn from all nations, but research itself is often conducted in teams that span countries. The most obvious of these are those associated with high energy physics, such as the experiments on the Large Hadron Collider, but international collaboration, even when it is just between two people, is very much the norm. The Institute itself has many international collaborators;

[5] https://rse.ac.uk/.
[6] http://www.de-rse.org/.

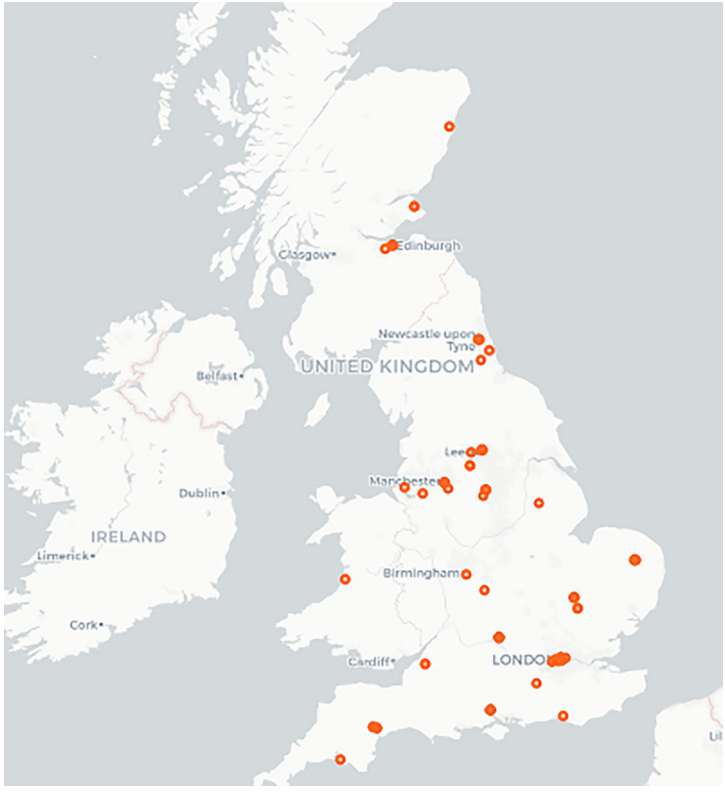

Fig. 2 Map of UK carpentry instructors affiliations (1/2018)

in addition to the RSE groups and Carpentries, the Institute works with groups like the Center for Trustworthy Scientific Cyberinfrastructure[7] in the US to tap into their expertise on security and resilience, Software Heritage[8] in France for software preservation, the Future of Research Communications and e-Scholarship[9] and Research Data Alliance[10] on software citation and credit, and the International Coalition on Science Gateways[11]. Working to understand the similarities and differences between how research software is treated in different countries also helps us to work together to change the research culture, for the benefit of all.

This is the ultimate goal of e-Science, as first defined nearly 20 years ago: to bring together teams from across disciplines, roles, and countries in collaboration to solve research's grand challenges. It is also the challenge of e-Science, which was perhaps misunderstood in those early years of the UK e-Science program, that to be successful it must concentrate on the people and knowledge transfer, more than on the hardware and software. The practice of research has changed substantially, it is our role as users and developers of research software to ensure that we continue to strive for research integrity and transparency over blind reproducibility and recognize the essential role of those who help make software reusable.

[7] https://trustedci.org/.
[8] https://www.softwareheritage.org/.
[9] https://www.force11.org/.
[10] https://www.rd-alliance.org/.
[11] http://www.icsciencegateways.org/.

References

1. Aleksic J, Alexa A, Attwood TK, Chue Hong N, Dahlö M, Davey R (2015) An open science peer review oath. F1000Research. https://doi.org/10.12688/f1000research.5686.2
2. Baxter R et al (2012) The research software engineer. In: Proceedings of the Digital Research Conference, Oxford

3. BBSRC and MRC (2014) Review of vulnerable skills and capabilities. https://www.mrc.ac.uk/documents/pdf/review-of-vulnerable-skills-and-capabilities/, last access: 1.2.2018
4. Begley CG, Ellis LM (2012) Drug development: Raise standards for preclinical cancer research. Nature 483(7391):531–533, https://doi.org/10.1038/483531a, PMID, 22460880
5. Brown CT (2017) How I learned to stop worrying and love the coming archivability crisis in scientific software. http://ivory.idyll.org/blog/2017-pof-software-archivability.html, last access: 1.2.2018
6. Collberg C, Proebsting TA (2016) Repeatability in computer systems research. Commun ACM 59(3):62–69, https://doi.org/10.1145/2812803
7. Coy P (2013) FAQ: Reinhart, Rogoff, and the Excel Error that changed history. Bloomberg. https://www.bloomberg.com/news/articles/2013-04-18/faq-reinhart-rogoff-and-the-excel-error-that-changed-history, last access: 1.2.2018
8. Crouch S, Chue Hong N, Hettrick S, Jackson M, Pawlik A, Sufi S, Parsons M (2013) The software sustainability institute: changing research software attitudes and practices. Comput Sci Eng 15(6):74–80, https://doi.org/10.1109/mcse.2013.133, last access: 1.2.2018
9. Enserink M (2017) Rethinking the dreaded r-word. Science 356(6342):998, https://doi.org/10.1126/science.356.6342.998
10. Gallego Llorente M, Jones ER, Eriksson A, Siska V, Arthur KW, Arthur JW, Curtis MC, Stock JT, Coltorti M, Pieruccini P, Stretton S, Brock F, Higham T, Park Y, Hofreiter M, Bradley DG, Bhak J, Pinhasi R, Manica A (2016) Erratum for the Report "Ancient Ethiopian genome reveals extensive Eurasian admixture in Eastern Africa" (previously titled "Ancient Ethiopian genome reveals extensive Eurasian admixture throughout the African continent"). Science 351(6275):aaf3945–aaf3945, https://doi.org/10.1126/science.aaf3945
11. Hettrick SJ et al (2014) UK Research Software Survey 2014, https://dx.doi.org/10.5281/zenodo.14809
12. Hinsen K (2017) Sustainable software and reproducible research: dealing with software collapse, http://blog.khinsen.net/posts/2017/01/13/sustainable-software-and-reproducible-research-dealing-with-software-collapse/, last access: 1.2.2018
13. Ioannidis JPA, Allison DB, Ball CA, Coulibaly I, Cui X, Culhane AC, van Noort V et al (2008) Repeatability of published microarray gene expression analyses. Nat Genet 41(2):149–155, https://doi.org/10.1038/ng.295
14. Katz DS (2017) Is software reproducibility possible and practical? https://danielskatzblog.wordpress.com/2017/02/07/is-software-reproducibility-possible-and-practical/, last access: 1.2.2018
15. Mann RP, Perna A, Ströbom D, Garnett R, Herbert-Read JE, Sumpter DJT, Ward AJW (2012) Retraction: Multi-scale inference of interaction rules in animal groups using bayesian model selection. PLoS Comput Biol 8(8), https://doi.org/10.1371/annotation/7bc3a37e-db82-4813-8242-7d34877125c5
16. Nangia U and Katz DS (2017) Track 1 Paper: Surveying the US National Postdoctoral Association regarding software use and training in research, https://doi.org/10.6084/m9.figshare.5328442.v3
17. National Academies of Sciences, Engineering, and Medicine (2016) Future Directions for NSF Advanced Computing Infrastructure to Support US Science and Engineering in 2017–2020. The National Academies Press, Washington, DC, https://doi.org/10.17226/21886
18. RCUK Review of e-Science (2009) Building a UK Foundation for the Transformative Enhancement of Research and Innovation. https://www.epsrc.ac.uk/newsevents/pubs/rcuk-review-of-e-science-2009-building-a-uk-foundation-for-the-transformative-enhancement-of-research-and-innovation/, last access: 1.2.2018
19. Reinhart C, Rogoff K (2010) Growth in a time of debt. National Bureau of Economic Research, https://doi.org/10.3386/w15639
20. Sufi S (2016) The Software Sustainability Institute Fellowship Programme: Supporting the Social Side of Research. In: Proceedings of the 4th Workshop on Sustainable Scientific Software: Practice and Experience. CEUR-WS, http://ceur-ws.org/Vol-1686/WSSSPE4_paper_27.pdf, last access: 1.2.2018
21. Sufi S, Peru G, Robinson J, Carr L, De Roure D, Goble C, Pawlik A (2014) Software in Reproducible Research. In: Proceedings of the 1st ACM SIGPLAN Workshop on Reproducible Research Methodologies and New Publication Models in Computer Engineering – TRUST '14. ACM Press, https://doi.org/10.1145/2618137.2618140
22. Wilson G (2016) Software Carpentry: Lessons Learned. F1000Research, https://doi.org/10.12688/f1000research.3-62.v2
23. Wilson G, Aruliah DA, Brown CT, Chue Hong NP, Davis M, Guy RT, Wilson P (2014) Best practices for scientific computing. PLoS Biol 12(1):e1001745, https://doi.org/10.1371/journal.pbio.1001745